

### **Remarks and Arguments**

Claims 1-45 have been presented for examination. Claims 1, 12, 23, 34 and 45 have been amended.

Paragraph 2 of the office communication states that claims 1, 13, 23, 34 and 45 have been rejected under 35 U.S.C. §102(e) as anticipated by U.S. Patent No. 6,650,619 (Flood.) However, paragraph 3 indicates that this latter patent may be incorrect and that the rejection may have been intended over U.S. Patent No. 6,377,984 (Najork) instead of the Flood patent as recited. During a telephone conversation on February 15, 2005 between Examiner Jaroenchonwanit and the undersigned, it was confirmed that the rejection in paragraph 2 was intended to be over Najork rather than Flood. Accordingly, the rejection over Najork will be discussed below.

The present invention concerns a method and apparatus for dividing dynamically allocated tasks among multiple concurrent threads in order to avoid expensive atomic operations that are normally required. In order to perform this division, each queue of a plurality of task queues is associated with a different ordered pair of the concurrent threads. One thread of the ordered pair is called the "enqueueer" of that queue and the other thread of the ordered pair is called the "dequeueer." Note that a thread may be the enqueueer of many task queues, but each queue has only one enqueueer. Similarly, another thread may be the dequeueer of many task queues, but each queue has only one dequeueer.

When one of the concurrent threads identifies a task to be performed, it pushes an identifier of that task onto one or more of the queues for which it is an enqueueer. Similarly, when a thread needs a new task to perform, it pops a task identifier from a queue from which it is the dequeueer and performs the identified task. Since there is only one thread that pushes task identifiers onto each queue and only one thread that pops those identifiers from it, the atomic operations and their associated performance costs are avoided.

The Najork reference discloses a web crawler that uses multiple queues for scheduling downloads from host computers that host the crawled web pages. The queues hold URLs of web pages that must be downloaded for searching. The crawler is designed so that it does not submit multiple parallel download requests to the same

host computer, which requests might cause the host computer to crash. To this end, the Najork crawler assigns host components to queues so that all URLs with the same host component are assigned to the same queue. Each queue is assigned a single thread that dequeues URLs from that queue. The dequeuing thread also enqueues any URLs that are discovered during processing of the downloaded documents. However, many threads enqueue URLs to a given queue. This happens because each queue is associated with one or more host components. Thus, any enqueueing thread that encounters a URL with one of those host components will enqueue URLs to that queue. This operation is explained at Najork, column 2, lines 55-68. Najork, Figure 4A, shows a single DEMUX 404 performing the enqueueing for several FIFO queues 406-0 to 406-n. Thus, it appears that the single process is enqueueing to each FIFO. However, this figure represents the process performed by each enqueueing thread. Thus the DEMUX procedure is actually performed by each enqueueing thread. See Najork, column 6, lines 38-43. Thus, multiple threads enqueue to each FIFO queue.

Consequently, the operation in Najork is not the same, as presently claimed, in which a single enqueueer thread enqueues all task IDs in a given queue. Claims 1, 12, 23, 34 and 45 have been amended to specifically point out this difference. For example, claim 1 has been amended, in lines 7-10, to recite "...wherein the execution threads operate so that only an enqueueer of a task queue adds entries to that task queue and only the dequeuer of a task queue removes entries from that task queue ...". This is in contrast to the Najork web crawler in which multiple threads enqueue to each FIFO queue. Consequently, amended claim 1 patentably distinguishes over the cited reference. Similar amendments have been made to claims 12, 23, 34 and 45 and, consequently, these claims distinguish over the Najork reference in the same manner.

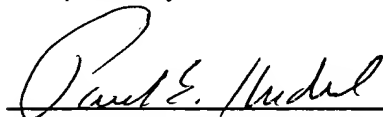
Claims 1-45 have been rejected under 35 U.S.C. §102(e) as anticipated by U.S. Patent No. 6,650,619 (Flood.) The examiner analogizes the task queues recited in the present claims with the work stealing queues shown in Flood Figures 4A-4C. In particular the examiner points to Figure 4B which shows a work stealing queue as it is being serviced by a single thread. In this arrangement, a single garbage collection (GC) thread both enqueues tasks (pushes tasks) onto the queue and dequeues tasks (pops tasks off) from the queue. However, as described in Flood, the queues are designed for

"work stealing." If any queue becomes empty, the thread associated with that queue can "steal" a task from another queue and add that task to its own queue (see Flood, column 7, lines 46-57 and Figure 4C). The mechanism for doing this is described the Arora article referenced at column 7, lines 12-15. Therefore, effectively each queue has more than one dequeuer: the thread that is associated with the queue and all other threads because these threads can steal from that queue. Thus, the Flood work-stealing queues do not achieve the advantages of the present invention. As discussed above independent claims 1, 12, 23, 34 and 45 have been amended to recite that "... only the dequeuer of a task queue removes entries from that task queue ..." That is clearly not true in Flood. Therefore, claims 1, 12, 23, 34 and 45 patentably distinguish over the cited Flood reference.

The remaining claims 2-11, 13-22, 24-33 and 35-44 are dependent, either directly or indirectly, on one of the independent claims and incorporate the limitations thereof. Therefore, they distinguish over the cited flood reference in the same manner as the independent claims.

In light of the forgoing amendments and remarks, this application is now believed in condition for allowance and a notice of allowance is earnestly solicited. If the examiner has any further questions regarding this amendment, he is invited to call applicants' attorney at the number listed below. The examiner is hereby authorized to charge any fees or direct any payment under 37 C.F.R. 1.17, 1.16 to Deposit Account number 02-3038.

Respectfully submitted



Paul E. Kudirka, Esq. Reg. No. 26,931

KUDIRKA & JOBSE, LLP

Customer Number 021127

Tel: (617) 367-4600 Fax: (617) 367-4656

Date: 2/18/05